Reflections on the Meltdown fix for FreeBSD

Konstantin Belousov `kib@freebsd.org`

April 14, 2018

`git src: 2018-04-06 18:01:51 +0300 ff44cd6`

Organizational

Questions

- Short questions in line
- Discussion after the blocks and at the end of the talk

Agenda

Talk Content

- Introduction
- What is Meltdown
  - Which CPU are vulnerable
  - How to check
- Page Table Isolation
- Kernel Entry
  - `sysenter` and `swapgs`
  - `iretq` and OS bugs
  - `NMI` and `MCE`
- Performance impact
- PCID
- i386: 4/4 UVA/KVA

What's wrong

What is Meltdown

- Speculative Execution
- Microarchitecture state leaks
- No U/S check

Disclosure Disaster

Image of the Sad Panda

Which CPUs are vulnerable

- Intel Cores: *yes*
- pre-Nehalem: (Pentium IV, Core2): I do not know
- Atoms: I suspect no
- AMD: no
- ARMs: yes for some Cortexes

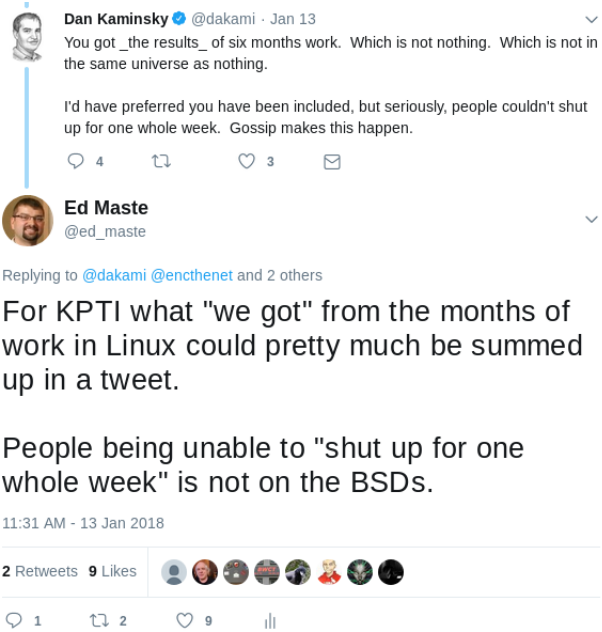
Test program

`https://github.com/dag-erling/meltdown`

Mitigating Meltdown: Page Table Isolation

- Developed for Linux as KAISER

Page Table Isolation: drama



Dan Kaminsky @dakami · Jan 13

You got _the results_ of six months work. Which is not nothing. Which is not in the same universe as nothing.

I'd have preferred you have been included, but seriously, people couldn't shut up for one whole week. Gossip makes this happen.

Ed Maste
@ed_maste

Replying to @dakami @encthenet and 2 others

For KPTI what "we got" from the months of work in Linux could pretty much be summed up in a tweet.

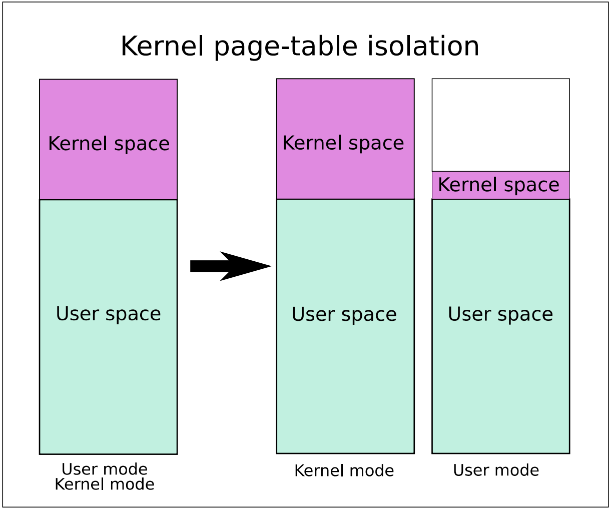
People being unable to "shut up for one whole week" is not on the BSDs.

11:31 AM - 13 Jan 2018

2 Retweets **9** Likes

Page Table Isolation

https://upload.wikimedia.org/wikipedia/commons/3/33/Kernel_page-table_isolation.svg



Page Table Isolation, Technical Details

Two page tables: user + trampoline vs. user + full kernel

User table

- User address space
- CPU system tables: GDT, IDT, TSS, LDT
- trampoline code
- minimal trampoline stack
- PCPU

Kernel table

- User address space: for copyout(9)
- Whole kernel text and data

Kernel Entry

Sysenter

- CPL and %rip
- OS duty: registers and *stack*
- AMD hack: `SWAPGS`

Rant

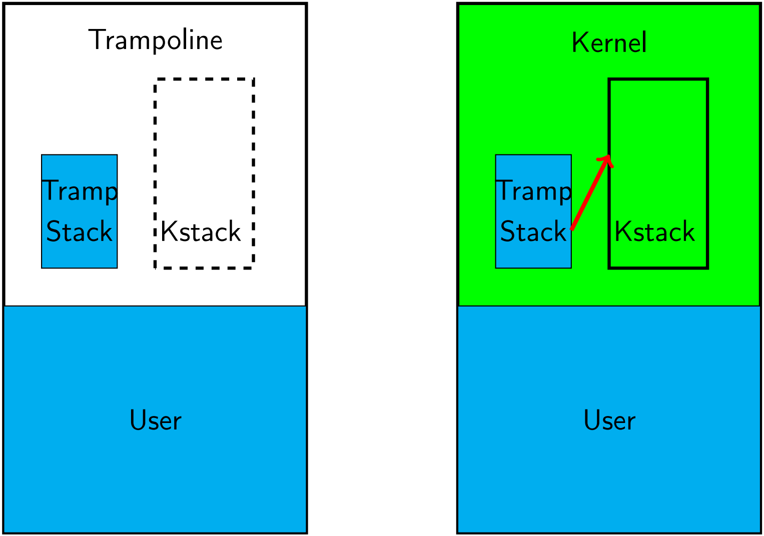
- OS bugs
- Special guest: `IRETQ`, Intel != AMD
  FreeBSD SA 15:21 amd64
- `NMI` and `MCE`

PTI Kernel Entry

- switch page table
- do it only when needed

- trampoline stack: copy frame to normal stack

PTI Kernel Entry: stack copy



Performance Impact

Reasons

- Global Pages no longer
- Full TLB flush on kernel->user
- Trampoline

getppid(2) timings

Syscall microbenchmark, wall clock time increase

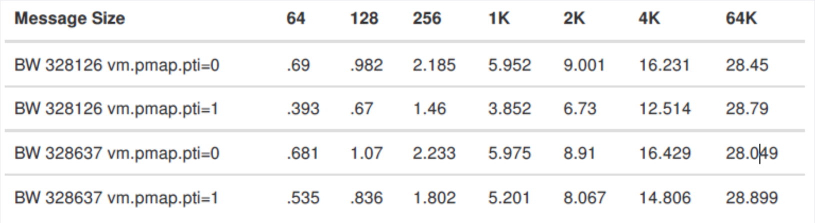
```
PTI on:              187.7% +/- 29.8653%
PTI on, using PCID:  119.7% +/- 21.5323%
```

Buildworld

```
real and user don't change at 95% confidence
sys increases by 3%
```

Performance impact: networking

Mellanox



Message Size	64	128	256	1K	2K	4K	64K
BW 328126 vm.pmap.pti=0	.69	.982	2.185	5.952	9.001	16.231	28.45
BW 328126 vm.pmap.pti=1	.393	.67	1.46	3.852	6.73	12.514	28.79
BW 328637 vm.pmap.pti=0	.681	1.07	2.233	5.975	8.91	16.429	28.049
BW 328637 vm.pmap.pti=1	.535	.836	1.802	5.201	8.067	14.806	28.899

PCID

Address Space Identifiers

- Pre-Meltdown Uses: optimize TLB flush on ctx switch
- Assign unique ID to full page table, user id = kernel id + 0x8000
- Switch PCID on kernel<->user switches
- Still full TLB flush on context switch. KVA in all kPCIDs.
- TLB Shutdown IPI: flush both user and kernel translations

i386

Still alive

- 3G UVA and 1G KVA: cannot link clang
- PTI ?
- Full 4G UVA and 4G KVA
- copyout(9) slow

References

- Intel 64 and IA-32 Architectures Software Developer Manuals, Volume 3
- AMD, AMD64 Architecture Programmer's Manual Volume 2: System Programming
- Meltdown paper `https://meltdownattack.com/meltdown.pdf`
- KAISER `https://lwn.net/Articles/738997/`
- FreeBSD wiki `https://wiki.freebsd.org/SpeculativeExecutionVulnerabilities`
- FreeBSD PoC `https://github.com/dag-erling/meltdown`
- PTI commit r328083
- PCID optimization r328470
- 4/4 i386 review `https://reviews.freebsd.org/D14633`

Questions

Ask Intel.